# The Support of Mobile-Awareness in Collaborative Groupware

**Keith Cheverst, Gordon Blair, Nigel Davies and Adrian Friday**

*Distributed Multimedia Research Group, Department of Computing, Lancaster University, Lancaster, UK*

**Abstract:** This paper explores the need for mobile-awareness in groupware designed for use in a mobile environment. The motivation for providing this additional form of awareness is that current groupware tends to assume a reliable and constant quality of communications that does not exist in a mobile environment. The implications of making this false assumption are twofold: first, systems that enforce certain ordering and reliability semantics across all operations can suffer drastic performance penalties; second, users are given insufficient awareness of (and control over) the effect of the unreliable communications environment on their collaboration. The support of mobile-awareness implies providing users with appropriate and sufficient information to adapt their behaviour (and the behaviour of the system) in response to changes in the quality of group communications available. To support the development of mobile-aware groupware we have built a flexible Quality of Service (QoS)-based group service capable of providing group-oriented feedback to application level services.

**Keywords:** Group service; Groupware; Mobile-awareness; Open distributed systems; Wireless communications; User interface design

## 1. Introduction

The main argument of this paper concerns the need for a new type of *awareness*, i.e. *mobile-awareness*, which provides group members with relevant information concerning the effect that the constraints imposed by their mobile environment might have on the group's collaboration. This approach couples and expands upon the notion of adaption [1] and the approach argued in [2], i.e. providing users with increased levels of awareness to deal with the problems of group collaboration in the presence of unreliable communications. In order to investigate some of the issues regarding the development of this class of groupware, we have experimented with providing varying degrees of mobile-awareness through our MOST (Mobile Open Systems Technologies) prototype groupware application using our own flexible QoS (or Quality-of-Service)-based group service.

The identification of mobile-awareness arose from the extensive body of work carried out during the MOST project [3]. This project worked closely with a REC (Regional Electricity Company) in order to investigate the potential for using portable and wireless technologies to improve collaboration between highly mobile field engineers in the safety-critical domain of the UK power distribution industry. It was anticipated that by improving collaboration, the operational efficiency of the REC could also be improved. Optimal efficiency is of prime importance to the REC because it has to manage the supply of electricity to approximately two million electricity consumers. Problems with the power distribution arise quite frequently and, when such problems leave consumers without supply, there is a strong financial incentive for the REC to return supply as soon as possible. A control centre is responsible for coordinating the work of field engineers and for maintaining an up-to-date view of the network state. For reasons of safety, it is crucial that such a view be maintained and that no inconsistencies regarding the current network state exist. For example, the control centre needs to be certain that a section of network has been "earthed" before instructing a field engineer to perform a repair on that section of network. In order to maintain the consistency of network views the control centre imposes a sequential ordering on all operations affecting the network.

When the MOST project started, the REC was in the process of changing over from a *push-to-talk*-based PMR (Private Mobile Radio) communications system to a more up-to-date connection-oriented PMR system. One implication of this change was that field engineers and control centre personnel would no longer be able to collaborate via a common broadcast channel. However, the new system was capable of supporting mobile data, and the REC was keen to investigate the extent to which the new communications infrastructure

could support enhanced collaboration between its workforce and the provision of mobile data services to its field engineers.

Following an extensive requirements-capture exercise, the MOST team developed a prototype distributed groupware application to enable multimedia collaboration between field engineers and control centre personnel. Because of the highly mobile nature of field engineers, the application was designed to run on handheld or highly portable computers using the companies wireless PMR system for group communications. This application was arguably the first collaborative mobile application ever built which was capable of adaption in a mobile environment [4].

The application was designed as an expandable toolkit comprising a number of modules including a shared GIS (Geographical Information System) module. This module supports both private and public (i.e. group) modes of working. In group mode, the module enables field engineers to perform spatially aware collaboration by supporting the display and annotation of network schematics across groups. Flexible WYSIWIS (or What You See Is What I See) is supported by the GIS module by virtue of its support for real-world (i.e. North-Easting) coordinates as opposed to actual screen coordinates. This approach enables engineers to view network schematics at different scales and locations but still receive shared annotations. Figure 1 illustrates the user interface to the GIS module and shows the various tools available for annotating network schematics.

The MOST application was evaluated by real end-users in a trial scenario using the wireless GSM service for communications. This evaluation provided a valuable set of implications regarding the development of distributed groupware in conjunction with mobile technologies.

## 2. The Impact of Mobile Communications on Distributed Groupware

Mobile communications implies the utilisation of different networking technologies in order to

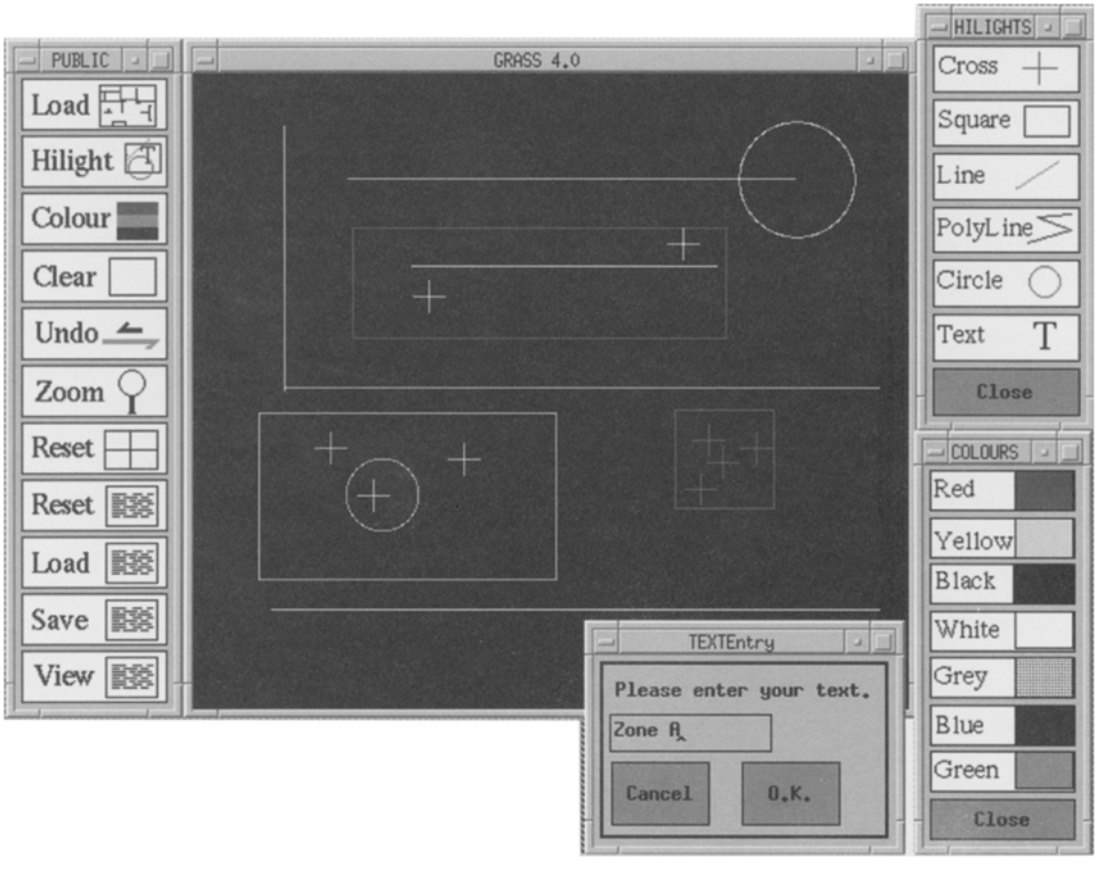


**Fig. 1.** Graphical user interface (GUI) to the Geographical Information System (GIS) module.

maintain network connectivity whilst mobile. For example, mobile computers can be either disconnected, weakly connected by low-speed wireless networks such as GSM, or fully connected by high-speed networks ranging from Ethernet to ATM.

The problem faced by developers when building mobile applications is that, when users roam between areas of different network infrastructure, rapid and massive fluctuations in the quality of service provided by the underlying communications infrastructure can occur. For example, a user might begin the day with their portable computer docked to a docking station with a high-bandwidth (i.e. 100 Mbps) ATM network link. Later on, the user may choose to undock their portable and move around their department whilst maintaining network connectivity through a lower bandwidth, local area RF (radio frequency)-based, network such as WaveLan (providing a maximum bandwidth of 2 Mbps). When leaving the department building, the user could still achieve network connectivity by utilising the wide-area but low-bandwidth (i.e. 9.6 Kbps) GSM service. However, whilst using this service the user might occasionally enter areas referred to as "coverage blackspots" and so lose network connectivity.

One of the MOST application's key requirements was the ability to operate over both heterogeneous networking and processing architectures. In particular, the application was required to be capable of switching between analogue PMR, GSM and wired Ethernet-based networks and also capable of running on both portable windows-based PCs and legacy-based UNIX platforms. For this reason, MOST used the open distributed processing (ODP)-based ANSAware [5] distributed systems toolkit as its development platform. ANSAware is a partial implementation of the Advanced Networked Systems Architecture (ANSA) [6] and has been influential in the specification of the reference model for ODP (RM-ODP) [7,8] by the International Standards Organization (ISO). In common with more recent open standards, such as CORBA (Common Object Request Broker Architecture) [9], RM-ODP is based around a client-server architecture and is strongly object-oriented.

The ODP model can be viewed and managed from a number of different viewpoints. Of most relevance to this paper is the computational viewpoint which is based on a location-independent *object-based* model of distributed systems. In this model, interacting entities are treated uniformly as *objects*, i.e. encapsulations of state and behaviour. Objects are accessed through *interfaces* and objects offering *services* (known as *servers*) are made available by *exporting* their *interface reference* to a database of interface references known as a *trader*. An object wishing to interact with a service (known as a *client*) must first *import* that server's interface reference. Once the client has the interface reference it can proceed to make an *invocation* on the server.

RM-ODP-based platforms provide application developers with a number of *abstractions* or *transparencies* for masking out various features of a distributed computation. Two examples of the transparencies provided are *network transparency* and *processing transparency*, which together enable systems to operate over a variety of machine/network configurations. Another transparency provided is *group transparency*. This gives the application programmer an abstraction for dealing with groups that hides specific group details such as the identity of individual group members.

Unfortunately, when developing mobile-aware groupware, these transparencies can hide from the programmer the details required for providing the user with mobile-awareness. For example, the strict enforcement of network and group transparencies makes it impossible for the application programmer to receive sufficient levels of feedback regarding changes in the quality of communications available to individual group members. The result of this is that mobile groupware developed using these transparencies tends to hide group communications problems from users and thus forces them to assume a constant level of communications quality.

In general, traditional distributed groupware systems (and the support services available to develop them) tend to hide details concerning the state of group communications from users and also assumes a constant level of communications QoS.

## 3. Mobile-Aware Distributed Groupware

The key to creating some form of compatibility between distributed groupware and mobile communications is mobile-aware groupware. Such groupware builds on the concept of awareness [10,11] to provide group members with feedback to make them fully aware (or rather as aware as they wish to be) of the effect of group communications on their collaboration. Such awareness should prevent group members from being forced to make (possibly false) assumptions regarding the current state of their connectivity with the rest of the group.

In order to investigate some of the implications of supporting mobile awareness, the MOST team developed an initial version of the MOST application which provided users a limited degree of mobile-awareness. This version of the application was evaluated by a group of end-users in a trial scenario. Based on the results of this evaluation, an enhanced version of the application was developed that tested the extent to which mobile-awareness could be supported. The following sections describe, in turn, the initial and enhanced versions of the MOST application.

### 3.1. The initial version of the MOST application

The initial version of the MOST application supported mobile-awareness by providing the end-user with an awareness of the current state of group connectivity. This form of awareness is provided via the graphical user interface (GUI) of the application's session manager module (shown in Fig. 2).

In more detail, the session manager's GUI comprises a number of scrollable areas in order to maximise the small display area available. The top left-hand side of the window contains icons representing the modules available to the user (the globe represents the GIS module). On the top right-hand side of the window is a scrollable column of icons representing engineers who can participate in a collaboration and below this there are icons for enabling members to be added and removed from the group membership. In the centre of the window is a row of "member" icons representing current group members. Under each member's icon is a column of "module" icons that represent the modules which that user is currently running.

Feedback regarding the state of connectivity within the group is achieved by colouring these "member" icons depending on the represented member's connectivity. For example, a disconnected group member's icon is displayed with a red background, while a connected group member's icon is displayed with a green background.

This initial version of the prototype MOST application was evaluated by members of the REC in a trial scenario in which a group of four users were asked to collaborate using the shared GIS module in order to locate damage to a "wiring pit" and carry out a repair. The group members communicated with one another using the wireless GSM service. During the collaboration, one member of the group had their phone switched off for a short period of time. This was done to mimic the impact on the group's collaboration of one of the group members entering a coverage black spot.

During the trial, the collaborating group were successfully able to use the application to affect the repair. In addition, the end-users found that the mobile-awareness information provided by the session manager enabled them to realise the period of time during which total group communication was not available and the identity of the member suffering disconnection from the group. This enabled the remainder of the group to adapt


**Fig. 2.** GUI to the initial session manager.

the coordination of their collaboration until the disconnected member regained connectivity with the rest of the group. However, the users involved in the trial felt that the application did not give them sufficient information regarding the way in which the constraints of operating in an unreliable mobile communications environment impacted upon their collaboration. More specifically, the following criticisms were raised.

***Insufficient temporal feedback:*** End-users were confused and frustrated when given no feedback or appreciation of the fact that establishing a connection to the rest of the group using the GSM service could take over 10 seconds and that significant delays could occur before their shared operations where received by all group members.

***Insufficient feedback on group consistency:*** Difficulties with communication and connectivity were not constant across all group members. For this reason, end-users wished for feedback regarding the identity of those particular group members who were unable to receive their shared operation. It is important to note that one of the application's requirements was to maintain a high level of data availability (even at the expense of possible inconsistencies between group members), and therefore the MOST application does not enforce either total or causal ordering. The potential for inconsistency was generally acceptable to engineers providing they were able to receive feedback should any inconsistencies arise. However, for certain operations engineers did require some form of consistency guarantee. For example, an engineer might require an operation to have atomic delivery semantics, i.e. to be received by a certain set of group members or none at all. Similarly, an engineer might require a shared operation to be received by a certain quorum of group members.

***Lack of support for managing the cost of group operations:*** One of the constraints imposed by a mobile communications infrastructure is cost. Depending on the service being used a cost may be charged on a per second or a per byte basis (or not at all). Where communication is provided by the company's own PMR system, managing the cost of calls between collaborating engineers is not usually an issue. However, the REC collaborating with the MOST project, despite using PMR, issued a number of its engineers with cellular phones in an attempt to overcome the problems of PMR coverage blackspots

## 3.2. The enhanced version of the MOST application

Following the criticisms described above, an enhanced version of the MOST application was developed in order to investigate some of the ways in which additional mobile-awareness information could be provided. The new version of the application was built to support a number of different forms of mobile-awareness. In order to enable the




**Fig. 3.** GUI for providing increased levels of awareness.

user to control the extent to which awareness information should be provided by the user interface, users are able to select different types of QoS in the knowledge that the system will only provide awareness information concerning the types of QoS selected by the user.

In more detail, the session manager's GUI (as shown in Fig. 3) was extended to provide:

1. a convenient and easy to use mechanism for associating QoS-based requirements with group operations, e.g. reliability, cost, ordering and temporal requirements
2. a clear and graphical method for receiving feedback should the communications environment cause any of the QoS requirements to be violated.

The user can specify QoS requirements using the smaller "guarantee" icons (i.e. those labelled O,C,G,T and Q) positioned beneath each member icon and each module icon. These icons represent from left to right: required ordering, maximum cost, required reliability, maximum delay and required quorum. The user can toggle each type of guarantee between active and inactive by simply clicking on the appropriate icon.

To provide the required level of flexibility, the GUI enables QoS requirements to be made against both individual group members and the entire collaborating group. For example, an engineer could specify the requirement that their next shared operation should be reliably received by the entire group but that only group member "Joe" needs to receive the operation within 2 seconds. This flexibility is important because certain group members, e.g. engaged in a monitoring process, might not have the same requirements as other group members.



### 3.3. The trade-off between complexity and ease-of-use

When considering the issue of mobile awareness one of the interesting issues that arises is the trade-off between ease of use and power/complexity. The mobile-aware user interface described in section 3.2 is clearly biased towards the power/complexity side of this trade-off. Such an interface is probably too complex to present to field engineers but did prove useful in highlighting the extreme level of mobile awareness that could be provided by a system and the potential for enabling end-users to control certain elements of the system.

Extensive evaluation is required to determine the optimum trade-off between the provision of awareness information and interface complexity. This trade-off is likely to be highly dependent on the collaborative task being performed and the preferences of individual users. For this reason, more specific, but less flexible, user interfaces are likely to be more appropriate for certain specific collaborative tasks. For example, once using the companies PMR system for collaboration, field engineers could be given mobile-awareness information relating only to group connectivity and the timeliness and reliability of group operations across all group members. In addition, by removing the ability for engineers to establish collaborating groups independently and launch application modules remotely, the user interface could be greatly simplified whilst still providing the core level of mobile-awareness required. However, the user interface provided to control centre personnel would need to provide the functionality to control groups and to monitor the ordering of operations and the consistency of shared views. This interface would therefore need to be more complex and demand from the user a greater level of technical expertise.

## 4. Support Services for Mobile-Aware Groupware

### 4.1. Overview

In order to support the development of mobile-aware distributed groupware, developers require novel and flexible group services. In particular, such services need to provide *flexible* group and network transparencies and also support the flow

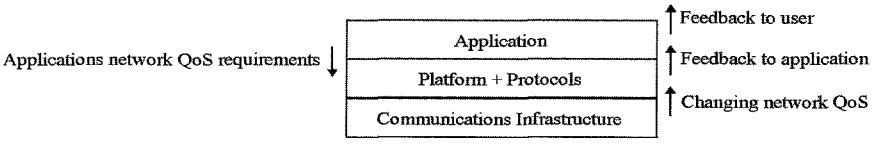


**Fig. 4.** Information flow required to support mobile applications.

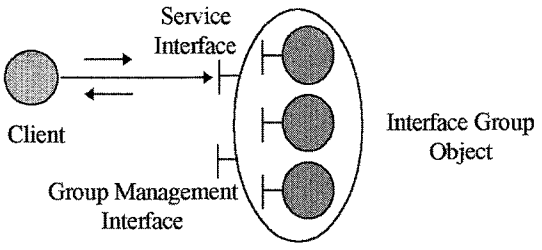


Fig. 5. The interface group abstraction.

of information between the application, the platform and the communications infrastructure (Fig. 4).

The group service described in [12] was built to support the development of mobile-aware groupware and was used to re-engineer the MOST groupware application. The group service is strongly based on *interface groups* [13], which are part of the ANSA model. An interface group provides a convenient abstraction over certain group issues and is basically a collection of interfaces which are only externally accessible through a single service interface and an associated group management interface. When an object invokes an operation on an interface group's service interface the operation is propagated to all group members contained in the interface group's membership. The group management interface is used to perform group membership tasks such as joining and removing members from an interface group's membership.

To illustrate the interface group abstraction, consider the shared GIS module. A client could propagate a GIS update operation to a group of GIS servers by simply invoking the update operation on the service interface of the corresponding interface group. Figure 5 illustrates a client invoking an operation on the service interface of an interface group comprising three members.

In a mobile environment, not all members of a collaborating group are likely to have the same quality of network connectivity, i.e. some members might enter a "coverage blackspot" whilst others have full connectivity. For this reason the group service enables the application programmer selectively to break group transparency by supporting the specification of QoS constraints that act on individual group members, in addition to those that act on the group as a whole.

## 4.2. Key properties

In order to support the development of mobile-aware groupware the group service possesses the following three key properties.

***Flexibility:*** One of the fundamental requirements of the group service is supporting flexible consistency between group members. As Greenberg and Marwood state [14], the degree of consistency required by any groupware system is application-dependent and therefore a "one size fits all" approach to manag-ing consistency is not suitable. In addition, the fundamental consistency/performance trade-off implies that strengthening the system's consistency guarantees results in a reduction of the system's responsiveness. Thus, in order to achieve optimum performance, it is important that supporting services only enforce the minimum level of consistency required by the application. Furthermore, because an application's consistency (or performance) requirements can change over time, support needs to be sufficiently flexible to react to any such changes.

To enable flexible consistency, the group service enables the relaxation of message ordering and reliability guarantees which, in a weakly connected environment, are extremely costly in terms of performance. Such a performance penalty is completely unnecessary if the shared operation being propagated does not require such strong guarantees.

In addition to message ordering and reliability requirements, the group service enables the application programmer to specify (over time) combinations of the following constraints:

1. The quorum of group members required to receive a shared operation; by reducing the quorum size, group invocations can succeed despite temporarily disconnected group members, although this obviously has an impact on group consistency.
2. The time-out period within which either the entire group or specific individual group members must acknowledge receipt of a shared operation; by increasing the length of time-out period, the programmer can effectively control the degree of *coupling* exhibited by certain parts of the system. It has been argued that flexible coupling is an important requirement in groupware systems [15]; however, for mobile groupware the requirement is even greater. For example, when group connectivity is good then a tightly coupled level of interaction can be maintained by the system. However, if the level of group connectivity becomes poor the level of coupling could be reduced. It is important to note that the setting of a time-out period can be can stipulated indirectly by the user through the user

interface, e.g. if the user requires feedback in the event of their next group action not reaching the group within 10 seconds. This means that the programmer does not need to have advanced knowledge of the timing characteristics of each operation.

3. The cost which the client is prepared to pay in order for either the entire group or specific group members to evaluate the shared operation. Note that the group service utilises the costing information (cost per byte, cost per second, etc.) provided by the underlying QEX protocol [16] in order to calculate the cost of propagating invocations to the group membership.

***Ability to provide feedback:*** In order to enable feedback to the application, the group service enables application programmers selectively to break group transparency by enabling them to associate specific QoS-based requirements with group updates. If, when propagating a group update, one or more of these requirements cannot be met then the service can provide feedback to the application.

***Ability to adapt:*** The group service is capable of performing intelligent adaption, i.e. tailoring its behaviour based on changes in the underlying communications infrastructure. For example, the group service can save resources by not attempting to propagate a group operation to any group member who is known to be currently disconnected and unreachable. To give a slightly more sophisticated example, consider a situation where a user has requested that his or her next group operation needs to be completed within 5 seconds. However, one of the group members has network connectivity provided by a GSM handset with a call set-up time of 10 seconds. In this situation, the response of the group service depends upon whether or not the group member equipped with GSM is connected at the time when the user issues the group operation. If the group member was not connected at the time then the group service should not attempt to propagate the operation to that member because the propagation could not occur within the specified time limit.

## 4.3. Supported application programming interface (API)

The API supported by the group service can be divided into three main areas of functionality.

***Group management:*** This part of the API is concerned with supporting interface group administration,



**Table 1.** Group management operations supported by the group service

| Operation | Description |
|---|---|
| *SetPopulation Policy* | Operation accepts two integer parameters representing the maximum and minimum of group members permitted |
| *Join* | Operation accepts as parameters a property name and value pairing for joining a compatible service interface to the group. Using this operation a client could, for example, instruct the group manager to join an interface instance to the interface group with the property name "GIS_Type" and value "Arc_Info" |
| *Leave* | Operation removes interface instances from the membership of the interface group by accepting a property name and value pairing for identifying the member to be removed |
| *ReturnNumMembers* | Operation returns the number of members in the interface group |

**Table 2.** Supported API for group-oriented QoS

| Operation | Description |
|---|---|
| *GetGroupQoS* | Operation returns a structure or profile showing which QoS guarantees a given client has chosen to stipulate |
| *SetGroupTimeQoS* | Operation enables a client to specify the time (in seconds) within which their subsequent group invocations need to be serviced |
| *SetGroupCostQoS* | Operation enables a client to specify the maximum cost (currently stipulated using units supported by the underlying platform) that should be incurred for servicing subsequent group invocations |
| *SetGroupQuorumQoS* | Operation enables a client to specify the quorum of group members required for servicing subsequent group invocations. The client can specify either an enumerated type, e.g. *all* or *majority*, or a cardinal value, e.g. two members |
| *SetGroupOrderingQoS* | Operation enables a client to specify (using a boolean value) whether group ordering is required |
| *SetGroupReliabilityQoS* | Operation enables a client to specify (using a boolean value) whether group reliability is required |

Table 3. Supported API for individual group member-oriented QoS

| Operation | Description |
|---|---|
| *GetGroupMembersQoS* | Operation returns a structure detailing the QoS guarantees that the client has chosen to specify for distributing operations to a specific member of the interface group |
| *SetGroupMembersTimeQoS* | Operation used to specify the time (in seconds) within which the client's group invocations need to be serviced by a specific group member |
| *SetGroupMembersCostQoS* | Operation used to specify the maximum cost that should be incurred for propagating subsequent group invocations to a specific group member |
| *SetGroupMembersOrderingQoS* | Operation used to specify (using a boolean value) whether a specific group member must receive invocations in order |
| *SetGroupMembersReliabilityQoS* | Operation used to specify (using a boolean value) whether or not a specific group member must receive group invocations |

e.g. adding members to and removing members from the interface group. The key operations supported by this part of the API are described in Table 1.

***Specification of QoS guarantees acting on the entire group membership:*** This part of the API (described in Table 2) enables a client to associate a combination of group-based QoS guarantees with an invocation on the interface group's service interface.

***Specification of QoS guarantees acting on individual group members:*** In order to enable clients to break group transparency completely, the group service enables a client to specify a combination of QoS guarantees on *individual* group members as opposed to the entire group. The operations that form this part of the API are described in Table 3. Note that the notion of providing a quorum guarantee is not applicable for individual group members.

## 5. Further Issues

In addition to the mobile awareness information provided by the MOST application, information on battery power constraints could also be made aware to users collaborating in a mobile environment. Depending on the type of portable end-system and wireless communications technology being used, the transmission of an invocation to the group service may consume a significant quantity of power. When this is the case, it may be appropriate to inform the user of the effect on battery power, associated with performing group operations.

It is clearly necessary to devise and develop new metaphors that can be used for making users aware of issues such as time, cost and power in a collaborative setting. Without such metaphors it is likely that users will feel that they are being presented with too much confusing information and this could have a detrimental effect on their ability to collaborate. This issue is of particular concern when developing interfaces that support mobile awareness for handheld systems with relatively small screen displays. More specifically, careful consideration needs to be given to questions such as:

1. Where is it best to locate awareness information in the UI?
2. To what extent do we enable users to specify what types of awareness they require?
3. How to provide users with the flexibility to specify the types of awareness that they are interested in whilst maintaining simplicity in the user interface?

The feasibility of supporting mobile-awareness relies on finding appropriate answers to the above questions. MOST has investigated these questions in one particular application domain, but it is clearly necessary to research and evaluate the usefulness of supporting mobile-awareness in other application domains, and such research is ongoing.

## 6. Summary

This paper has examined the problems associated with using distributed groupware in a mobile communications environment. The fundamental problem is that current groupware tends to assume a reliable and constant quality of group communications which, in a mobile environment, simply does not exist. The implications of making this false assumption are twofold. First, systems that enforce certain ordering and reliability semantics across all operations can suffer drastic performance penalties. Second, users will not be given sufficient awareness of (and control over) the effect of the

unreliable communications environment on their collaboration.

A new class of mobile-aware groupware is required in order to provide collaborating users with an awareness of the way in which the constraints imposed by the communications environment might affect group collaboration. Once presented with this awareness, users should be able to intelligently adapt their collaboration in response to changes in the level of group communications.

However, the problem with building mobile-aware distributed groupware using existing development tools and services, such as those based on RM-ODP, is that they mask the programmer from low-level networking and group-based details. This is a problem because in order to provide feedback to the user, applications require feedback from lower-level services regarding changes in group connectivity. In order to address this need, a flexible RM-ODP QoS-based group service has been developed that provides the kind of feedback required by mobile-aware groupware systems. In addition to providing feedback, the group service also enables the application programmer to control the system's performance/consistency trade-off by changing certain group propagation strategies, i.e. by altering group quorum and ordering requirements.

## References


1. Davies N, Cheverst K, Friday A et al. Supporting adaptive services in a heterogeneous mobile environment. In: Proceedings of Workshop on Mobile Computing Systems and Applications (MCSA), Santa Cruz, Calif. IEEE Computer Society Press, Los Alamitos, Calif., 1994
2. Dix A. Cooperation without (reliable) communication: interfaces for mobile applications. Distributed Systems Engineering 1995; 3: 171–181
3. Davies N, Friday A, Blair G, et al. Mobile open systems technologies for the utilities industries. In: Dix A (ed) Remote cooperation: CSCW for mobile and tele-workers, Springer, Berlin Heidelberg New York, 1995: 145–166
4. Davies N, Cheverst K, Friday A et al. Distributed systems support for adaptive mobile applications. ACM Mobile Networks and Applications (special issue on Mobile Computing – System Services) 1996; 4: 399–408
5. APM Ltd. An introduction to ANSAware 4.0. Architecture Projects Management Ltd, Cambridge, UK, 1992.
6. APM Ltd. The ANSA reference manual, release 01.00. Architecture Projects Management Ltd, Cambridge, UK, 1989
7. ISO/IEC 10746-2. ITU recommendation X.902, open distributed processing – reference model – part 2. Foundations. International Standards Organization, 1995
8. ISO/IEC 10746-3. ITU recommendation X.903, open distributed processing – reference model – part 3. Architecture. International Standards Organization, 1995
9. Object Management Group. The Common Object Request Broker: architecture and specification, revision 2.2, 1998. Available at: http://www.omg.org/corba/corbiiop.htm
10. Dourish P. Awareness and coordination in shared workspace. In: Proceedings of the ACM CSCW'92 Conference on Computer Supported Cooperative Work. Toronto, Canada. ACM Press, New York, 1992: 107–114
11. Lauwers J, Lantz K. Collaboration awareness in support of collaboration transparency. In: Proceedings of the ACM SIGCHI'90 Conference on Human Factors in Computing Systems. Seattle, Washington. ACM Press, New York, 1990: 303–211
12. Cheverst K, Davies N, Friday A et al. Services to support consistency in mobile collaborative applications. In: Proceedings of the 3rd International Workshop on Services in Distributed Networked Environments (SDNE). Macao, China. IEEE Computer Society Press, Los Alamitos, Calif., 1996: 27–34
13. Oskiewicz E, Warne J, Olsen M. A model for interface groups. APM/TR.009.00. Architecture Projects Management Ltd, Cambridge, UK, 1990
14. Greenberg S, Marwood D. Real time groupware as a distributed system: concurrency control and its effect on the interface. In: Proceedings of the ACM CSCW'94 Conference on Computer Supported Cooperative Work. Chapel Hill, N. C. ACM Press, New York, 1994: 22–26
15. Dewan P. Flexible user interface coupling in collaborative systems. In: Proceedings of the ACM CHI'91 Conference on Human Factors in Computing Systems. New Orleans, La. ACM Press, New York, 1991: 41–48
16. Friday A, Cheverst K, Davies N et al. Extensions to ANS Aware for advanced mobile applications. In: Proceedings of the International Conference on Distributed Platforms. Dresden. Chapman and Hall, London, 1996: 29–43



*Correspondence to:* Keith Cheverst, Distributed Multimedia Research Group, Department of Computing, Lancaster University, Lancaster LA1 4YR, UK. Email: kc@comp.lancs.ac.uk